Standard Operating Procedure (SOP)

Pharma ERP Application - Deployment Guide

Document Version: 2.0
Date: January 6, 2026
Project: Embellish - Pharma ERP System
Deployment URL: https://pharma.wizoneit.com
API URL: https://pharma.wizoneit.com/api

Table of Contents

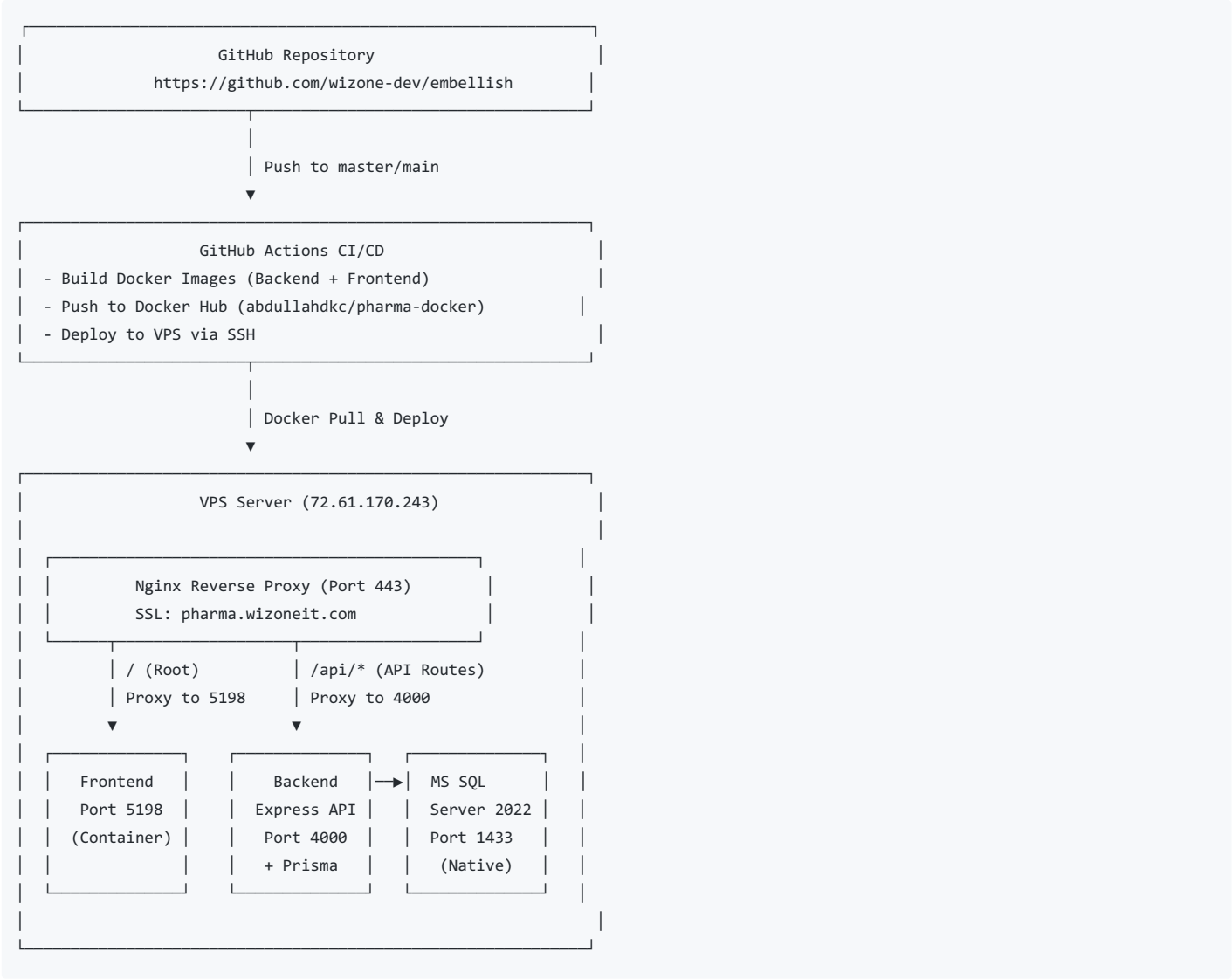
---

1. Overview

This SOP documents the complete deployment process for the Pharma ERP application from development to production. The application consists of:

- **Frontend:** React + TypeScript + Vite (Dockerized on Nginx)
- **Backend:** Node.js + Express + TypeScript + Prisma ORM (Dockerized on Node)
- **Database:** Microsoft SQL Server 2022 (Native on VPS)
- **Deployment:** Docker containers on VPS
- **CI/CD:** GitHub Actions
- **Domain:** pharma.wizoneit.com

Architecture



---

2. Prerequisites

Required Software

- **Git:** Version control
- **Docker:** v28+ and Docker Compose v1.29+
- **Node.js:** v20+ (for local development)
- **Microsoft SQL Server:** 2022 or 2019 (for production)
- **Prisma CLI:** `npm install -g prisma`
- **Code Editor:** VS Code recommended

Required Accounts

- **GitHub Account:** For repository hosting
- **Docker Hub Account:** For container registry
  - Username: `abdullahkhan9905`
  - Repository: `abdullahdkc/pharma-docker`
- **VPS Server:** Ubuntu 22.04 LTS
  - IP: `72.61.170.243`
  - User: `deployer`

Access Requirements

- SSH access to VPS
- GitHub repository access
- Docker Hub repository access

---

3. Local Development Setup

3.1 Clone Repository

```
# Clone the repository
git clone https://github.com/wizone-dev/embellish.git
cd embellish

# Repository structure
embellish/
├── .github/
│   └── workflows/
│       └── deploy.yml
├── backend/
│   ├── Dockerfile
│   ├── package.json
│   └── src/
├── frontend/
│   ├── Dockerfile
│   ├── package.json
│   └── src/
├── docker-compose.yml
└── README.md
```

3.2 Frontend Development

```
# Navigate to frontend
cd frontend

# Install dependencies
npm install

# Run development server
npm run dev
# Access: http://localhost:5173
```

3.3 Backend Development

```
# Navigate to backend
cd backend

# Install dependencies
npm install

# Install Prisma CLI globally (if not installed)
npm install -g prisma

# Create .env file for database connection
cat > .env << EOF
PORT=4000
NODE_ENV=development

# Microsoft SQL Server Connection
DATABASE_URL="sqlserver://localhost:1433;database=embellish_dev;user=sa;password=YourStrong@Passw0rd;encrypt=true;trustServerCertifi

# JWT Configuration
JWT_SECRET=dev_secret_key_change_in_production
JWT_EXPIRES_IN=12h

# CORS
CORS_ORIGIN=http://localhost:5173
EOF

# Generate Prisma Client
npx prisma generate

# Run database migrations
npx prisma migrate dev --name init

# (Optional) Seed database
npx prisma db seed

# Run development server
npm run dev
# Access: http://localhost:4000

# Test API endpoints
curl http://localhost:4000/api/health
```

Prisma Schema Example (`backend/prisma/schema.prisma`):

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "sqlserver"
  url      = env("DATABASE_URL")
}

model User {
  id        String   @id @default(uuid())
  email     String   @unique
  name      String?
  password  String
  role      String   @default("user")
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@map("users")
}

model Product {
  id          String   @id @default(uuid())
  name        String
  description String?
  sku         String   @unique
  price       Decimal  @db.Decimal(10, 2)
  quantity    Int
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  @@map("products")
}
```

---

4. Docker Image Build & Push

4.1 Build Docker Images Locally

```
# Navigate to project root
cd "d:\Telegram Desktop\embellish 2\embellish"

# Build backend image
docker build -t abdullahdkc/pharma-docker:backend ./backend

# Build frontend image
docker build -t abdullahdkc/pharma-docker:frontend ./frontend

# Verify images
docker images | findstr pharma
```

Build Output:

- Backend build time: ~30 seconds
- Frontend build time: ~8 minutes (includes TypeScript compilation)

4.2 Test Images Locally

```
# Run frontend container
docker run -d -p 5198:80 --name frontend-test abdullahdkc/pharma-docker:frontend

# Test in browser
# Open: http://localhost:5198

# Stop and remove test container
docker stop frontend-test
docker rm frontend-test
```

4.3 Push to Docker Hub

```
# Login to Docker Hub
docker login
# Username: abdullahkhan9905
# Password: [Enter your Docker Hub password]

# Push backend image
docker push abdullahdkc/pharma-docker:backend

# Push frontend image
docker push abdullahdkc/pharma-docker:frontend
```

Verification: Visit: https://hub.docker.com/r/abdullahdkc/pharma-docker/tags

---

5. VPS Server Setup

5.1 Initial Server Connection

```
# SSH to VPS
ssh deployer@72.61.170.243
# Password: AiLabs@2026
```

5.2 Install Docker (If Not Installed)

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Add user to docker group
sudo usermod -aG docker $USER

# Install Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/d
sudo chmod +x /usr/local/bin/docker-compose

# Verify installation
docker --version
docker-compose --version
```

Expected Output:

```
Docker version 28.2.2, build 28.2.2-0ubuntu1~22.04.1
docker-compose version 1.29.2, build unknown
```

5.3 Install Nginx (If Not Installed)

```
# Install Nginx
sudo apt install nginx -y

# Verify installation
nginx -v

# Check status
sudo systemctl status nginx
```

5.4 Install Certbot for SSL

```
# Install Certbot
sudo apt install certbot python3-certbot-nginx -y

# Verify installation
certbot --version
```

5.5 Install Microsoft SQL Server

```
# Import Microsoft GPG key
curl https://packages.microsoft.com/keys/microsoft.asc | sudo tee /etc/apt/trusted.gpg.d/microsoft.asc

# Add Microsoft SQL Server repository
curl https://packages.microsoft.com/config/ubuntu/22.04/mssql-server-2022.list | sudo tee /etc/apt/sources.list.d/mssql-server-2022.

# Update package list
sudo apt update

# Install SQL Server
sudo apt install -y mssql-server

# Configure SQL Server (choose Developer or Express edition)
sudo /opt/mssql/bin/mssql-conf setup
# Select: 2) Developer (free, no production use rights)
# Enter SA password: YourStrong@Passw0rd123

# Verify SQL Server is running
systemctl status mssql-server

# Install SQL Server command-line tools
curl https://packages.microsoft.com/config/ubuntu/22.04/prod.list | sudo tee /etc/apt/sources.list.d/mssql-release.list
sudo apt update
sudo ACCEPT_EULA=Y apt install -y mssql-tools18 unixodbc-dev

# Add tools to PATH
echo 'export PATH="$PATH:/opt/mssql-tools18/bin"' >> ~/.bashrc
source ~/.bashrc

# Test connection
sqlcmd -S localhost -U sa -P "YourStrong@Passw0rd123" -C
# Type: SELECT @@VERSION;
# Type: GO
# Type: EXIT
```

Expected Output:

```
Microsoft SQL Server 2022 (RTM) - 16.0.1000.6
```

5.6 Create Database

```
# Connect to SQL Server
sqlcmd -S localhost -U sa -P "YourStrong@Passw0rd123" -C

# Create database (in sqlcmd prompt)
CREATE DATABASE embellish_prod;
GO

# Create application user
USE embellish_prod;
GO

CREATE LOGIN pharma_user WITH PASSWORD = 'Pharma@SecurePass2026';
GO

CREATE USER pharma_user FOR LOGIN pharma_user;
GO

ALTER ROLE db_owner ADD MEMBER pharma_user;
GO

EXIT
```

Verify Database:

```
sqlcmd -S localhost -U pharma_user -P "Pharma@SecurePass2026" -d embellish_prod -C
SELECT DB_NAME();
GO
EXIT
```

---

6. Application Deployment

6.1 Clone Repository on VPS

```
# Navigate to web directory
cd /var/www/htdocs

# Backup existing setup (if any)
sudo cp pharma/docker-compose.yml ~/docker-compose-backup.yml

# Remove old structure
sudo rm -rf pharma

# Clone fresh repository
sudo git clone https://github.com/wizone-dev/embellish.git pharma

# Navigate to project
cd pharma
```

6.2 Create Environment File

```
# Create .env file with all configuration
sudo nano .env
```

`.env` File Content:

```
# Node Environment
NODE_ENV=production
PORT=4000

# Microsoft SQL Server Connection
# Format: sqlserver://HOST:PORT;database=DB_NAME;user=USER;password=PASSWORD;encrypt=true;trustServerCertificate=true
DATABASE_URL="sqlserver://172.17.0.1:1433;database=embellish_prod;user=pharma_user;password=Pharma@SecurePass2026;encrypt=true;trust

# JWT Configuration
JWT_SECRET=your_super_secret_jwt_key_production_2026_change_this
JWT_EXPIRES_IN=12h

# CORS Configuration
CORS_ORIGIN=https://pharma.wizoneit.com

# Frontend API URL
VITE_API_URL=https://pharma.wizoneit.com/api
```

Note: `172.17.0.1` is the Docker host gateway IP (allows containers to connect to host services like SQL Server)

Save: `Ctrl+X`, then `Y`, then `Enter`

6.3 Configure Docker Compose (Full Stack)

```
# Edit docker-compose.yml
sudo nano docker-compose.yml
```

docker-compose.yml (Backend + Frontend):

```
version: "3.8"

services:
  backend:
    image: abdullahdkc/pharma-docker:backend
    container_name: backend
    restart: always
    ports:
      - "4000:4000"
    environment:
      - NODE_ENV=${NODE_ENV}
      - PORT=${PORT}
      - DATABASE_URL=${DATABASE_URL}
      - JWT_SECRET=${JWT_SECRET}
      - JWT_EXPIRES_IN=${JWT_EXPIRES_IN}
      - CORS_ORIGIN=${CORS_ORIGIN}
    extra_hosts:
      - "host.docker.internal:172.17.0.1"

  frontend:
    image: abdullahdkc/pharma-docker:frontend
    container_name: frontend
    restart: always
    ports:
      - "5198:80"
    depends_on:
      - backend
    environment:
      - VITE_API_URL=${VITE_API_URL}
```

Save: `Ctrl+X`, then `Y`, then `Enter`

6.4 Login to Docker Hub on VPS

```
# Login to Docker Hub
sudo docker login
# Username: abdullahkhan9905
# Password: [Enter Docker Hub password]
```

6.5 Run Database Migrations

Before starting containers, run Prisma migrations to set up the database schema.

```
# Navigate to project directory
cd /var/www/htdocs/pharma/backend

# Install dependencies (if needed)
npm install

# Load environment variables
export $(cat ../.env | xargs)

# Generate Prisma Client
npx prisma generate

# Run migrations
npx prisma migrate deploy

# (Optional) Seed initial data
npx prisma db seed

# Verify database tables
sqlcmd -S 172.17.0.1 -U pharma_user -P "Pharma@SecurePass2026" -d embellish_prod -C
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE='BASE TABLE';
GO
EXIT
```

Expected Output:

```
users
products
_prisma_migrations
```

6.6 Pull and Run Containers

```
# Pull both images
sudo docker-compose pull

# Start all containers
sudo docker-compose up -d

# Verify containers are running
sudo docker ps
```

Expected Output:

```
CONTAINER ID   IMAGE                                COMMAND                  PORTS                    NAMES
341ccb97d3e5   abdullahdkc/pharma-docker:backend    "docker-entrypoint.s…"   0.0.0.0:4000->4000/tcp   backend
307dc6b5adae   abdullahdkc/pharma-docker:frontend   "/docker-entrypoint.…"   0.0.0.0:5198->80/tcp     frontend
```

6.7 Test Application

```
# Test backend API
curl http://localhost:4000/api/health
# Expected: {"status":"ok","timestamp":"2026-01-06T..."}

# Test backend with headers
curl -I http://localhost:4000/api/health
# Expected: HTTP/1.1 200 OK

# Test frontend
curl -I http://localhost:5198
# Expected: HTTP/1.1 200 OK

# Check backend logs
sudo docker logs backend --tail 50

# Check frontend logs
sudo docker logs frontend --tail 50
```

---

7. SSL & Domain Configuration

7.1 Configure Nginx Reverse Proxy

```
# Create Nginx configuration
sudo nano /etc/nginx/sites-available/pharma.wizoneit.com
```

Nginx Configuration (Backend + Frontend):

```
server {
    listen 80;
    server_name pharma.wizoneit.com;

    # Backend API Proxy
    location /api/ {
        proxy_pass http://localhost:4000/api/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # CORS headers (if needed)
        add_header 'Access-Control-Allow-Origin' 'https://pharma.wizoneit.com' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DELETE, OPTIONS' always;
        add_header 'Access-Control-Allow-Headers' 'Content-Type, Authorization' always;

        # Handle preflight requests
        if ($request_method = 'OPTIONS') {
            return 204;
        }
    }

    # Frontend Proxy
    location / {
        proxy_pass http://localhost:5198;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Save: `Ctrl+X`, then `Y`, then `Enter`

7.2 Enable Site

```
# Enable the site
sudo ln -s /etc/nginx/sites-available/pharma.wizoneit.com /etc/nginx/sites-enabled/

# Test Nginx configuration
sudo nginx -t

# Reload Nginx
sudo systemctl reload nginx
```

7.3 Configure SSL with Let's Encrypt

```
# Obtain SSL certificate
sudo certbot --nginx -d pharma.wizoneit.com

# Follow prompts:
# 1. Enter email address
# 2. Agree to terms
# 3. Choose redirect option (recommended)
```

Certbot will automatically:

- Generate SSL certificates
- Modify Nginx configuration
- Set up auto-renewal

7.4 Fix Port Configuration (If Needed)

If you encounter "Bad Gateway" errors, check if another Nginx config is interfering:

```
# Search for conflicting port configurations
sudo grep -r "5173" /etc/nginx/sites-available/

# If found, update to correct port (5198)
sudo nano /etc/nginx/sites-available/wizoneit

# Find and replace:
# proxy_pass http://127.0.0.1:5173/;
# WITH:
# proxy_pass http://127.0.0.1:5198/;

# Alternative: Use sed command
sudo sed -i 's|proxy_pass http://127.0.0.1:5173/;|proxy_pass http://127.0.0.1:5198/;|g' /etc/nginx/sites-available/wizoneit

# Test and reload
sudo nginx -t
sudo systemctl reload nginx
```

7.5 Verify Deployment

```
# Test frontend HTTPS
curl -I https://pharma.wizoneit.com
# Expected: HTTP/2 200

# Test backend API HTTPS
curl https://pharma.wizoneit.com/api/health
# Expected: {"status":"ok",...}

# Test backend API endpoint
curl https://pharma.wizoneit.com/api/users
# Expected: JSON array or authentication error
```

Access Points:

- Frontend: https://pharma.wizoneit.com
- Backend API: https://pharma.wizoneit.com/api

---

8. CI/CD Pipeline Setup

8.1 GitHub Workflow Configuration

The workflow file is already created at `.github/workflows/deploy.yml`:

```
name: Build and Deploy to Docker Hub

on:
```

```yaml
  push:
    branches:
      - main
      - master
  workflow_dispatch:

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: abdullahkhan9905
          password: ${{ secrets.DOCKER_PASSWORD }}

      - name: Build and push backend
        uses: docker/build-push-action@v5
        with:
          context: ./backend
          push: true
          tags: |
            abdullahdkc/pharma-docker:backend
            abdullahdkc/pharma-docker:backend-${{ github.sha }}
          cache-from: type=registry,ref=abdullahdkc/pharma-docker:backend-buildcache
          cache-to: type=registry,ref=abdullahdkc/pharma-docker:backend-buildcache,mode=max

      - name: Build and push frontend
        uses: docker/build-push-action@v5
        with:
          context: ./frontend
          push: true
          tags: |
            abdullahdkc/pharma-docker:frontend
            abdullahdkc/pharma-docker:frontend-${{ github.sha }}
          cache-from: type=registry,ref=abdullahdkc/pharma-docker:frontend-buildcache
          cache-to: type=registry,ref=abdullahdkc/pharma-docker:frontend-buildcache,mode=max

  deploy-to-vps:
    needs: build-and-push
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main' || github.ref == 'refs/heads/master'

    steps:
      - name: Deploy to VPS via SSH
        uses: appleboy/ssh-action@v1.0.3
        with:
          host: ${{ secrets.VPS_HOST }}
          username: ${{ secrets.VPS_USER }}
          key: ${{ secrets.VPS_SSH_KEY }}
          port: ${{ secrets.VPS_PORT || 22 }}
          script: |
            cd /var/www/htdocs/pharma

            # Login to Docker Hub
            echo "${{ secrets.DOCKER_PASSWORD }}" | sudo docker login -u abdullahkhan9905 --password-stdin

            # Pull latest images
```

```
            sudo docker-compose pull

            # Run database migrations
            cd backend
            export $(cat ../.env | xargs)
            npx prisma migrate deploy
            cd ..

            # Restart containers
            sudo docker-compose up -d

            # Clean up old images
            sudo docker image prune -af --filter "until=48h"

            # Show status
            sudo docker-compose ps
            sudo docker logs backend --tail 20
```

8.2 Configure GitHub Secrets

1. Go to GitHub repository: https://github.com/wizone-dev/embellish
2. Click **Settings → Secrets and variables → Actions**
3. Click **New repository secret**

Add the following secrets:

Secret Name	Value	Description
`DOCKER_PASSWORD`	[Your Docker Hub password]	Docker Hub authentication
`VPS_HOST`	`72.61.170.243`	VPS server IP address
`VPS_USER`	`deployer`	SSH username
`VPS_SSH_KEY`	[Private SSH key]	SSH private key for authentication
`VPS_PORT`	`22`	SSH port (optional, defaults to 22)

8.3 Generate SSH Key for GitHub Actions

On your VPS:

```
# Generate SSH key pair
ssh-keygen -t ed25519 -C "github-actions" -f ~/.ssh/github_actions_key -N ""

# Add public key to authorized_keys
cat ~/.ssh/github_actions_key.pub >> ~/.ssh/authorized_keys

# Set correct permissions
chmod 600 ~/.ssh/authorized_keys
chmod 700 ~/.ssh

# Display private key (copy this to GitHub secret)
cat ~/.ssh/github_actions_key
```

Copy the entire output (including `-----BEGIN OPENSSH PRIVATE KEY-----` and `-----END OPENSSH PRIVATE KEY-----`) and add it as the `VPS_SSH_KEY` secret in GitHub.

8.4 Test GitHub Actions

```
# Make a small change locally
echo "# Test deployment" >> README.md

# Commit and push
git add README.md
git commit -m "Test CI/CD pipeline"
git push origin master
```

Go to GitHub → Your repository → **Actions** tab to monitor the workflow.

---

9. Monitoring & Maintenance

9.1 Container Management

```
# View running containers
sudo docker ps

# View all containers (including stopped)
sudo docker ps -a

# View logs for all containers
sudo docker-compose logs -f

# View backend logs
sudo docker-compose logs -f backend
sudo docker logs backend --tail 100

# View frontend logs
sudo docker-compose logs -f frontend
sudo docker logs frontend --tail 100

# Restart all containers
sudo docker-compose restart

# Restart specific container
sudo docker-compose restart backend
sudo docker-compose restart frontend

# Stop all containers
sudo docker-compose stop

# Start all containers
sudo docker-compose start

# Remove all containers
sudo docker-compose down

# Access backend container shell
sudo docker exec -it backend sh

# Access frontend container shell
sudo docker exec -it frontend sh
```

9.2 Update Application

Manual Update:

```
# SSH to VPS
ssh deployer@72.61.170.243

# Navigate to project
cd /var/www/htdocs/pharma

# Pull latest code (if needed for migrations)
git pull origin master

# Pull latest Docker images
sudo docker-compose pull

# Run database migrations
cd backend
export $(cat ../.env | xargs)
npx prisma migrate deploy
cd ..

# Restart all containers
sudo docker-compose up -d

# Verify both containers are running
sudo docker ps

# Check logs
sudo docker logs backend --tail 50
sudo docker logs frontend --tail 50

# Test backend API
curl https://pharma.wizoneit.com/api/health

# Test frontend
curl -I https://pharma.wizoneit.com
```

Automatic Update via CI/CD:

- Simply push changes to `master` or `main` branch
- GitHub Actions will automatically build, push, and deploy

9.3 Database Backup & Restore

Backup MS SQL Server Database:

```
# Method 1: Using sqlcmd (Full Backup)
sqlcmd -S localhost -U pharma_user -P "Pharma@SecurePass2026" -d embellish_prod -C -Q "BACKUP DATABASE [embellish_prod] TO DISK = N'

# Method 2: Export data as SQL script
sqlcmd -S localhost -U pharma_user -P "Pharma@SecurePass2026" -d embellish_prod -C -Q "SELECT * FROM users" -o backup_users_$(date +

# Method 3: Using Prisma (Schema + Data)
cd /var/www/htdocs/pharma/backend
npx prisma db pull
npx prisma db seed
```

Restore MS SQL Server Database:

```
# Restore from backup file
sqlcmd -S localhost -U sa -P "YourStrong@Passw0rd123" -C -Q "RESTORE DATABASE [embellish_prod] FROM DISK = N'/var/opt/mssql/backup/e

# Or reset and recreate using Prisma
cd /var/www/htdocs/pharma/backend
npx prisma migrate reset --force
npx prisma migrate deploy
npx prisma db seed
```

Automated Daily Backup Script:

```
# Create backup script
sudo nano /usr/local/bin/backup-pharma-db.sh
```

Script content:

```
#!/bin/bash
BACKUP_DIR="/var/opt/mssql/backup"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="$BACKUP_DIR/embellish_backup_$DATE.bak"

# Create backup
sqlcmd -S localhost -U pharma_user -P "Pharma@SecurePass2026" -d embellish_prod -C -Q "BACKUP DATABASE [embellish_prod] TO DISK = N'

# Delete backups older than 7 days
find $BACKUP_DIR -name "embellish_backup_*.bak" -mtime +7 -delete

echo "Backup completed: $BACKUP_FILE"
```

```
# Make executable
sudo chmod +x /usr/local/bin/backup-pharma-db.sh

# Add to crontab (daily at 2 AM)
(crontab -l 2>/dev/null; echo "0 2 * * * /usr/local/bin/backup-pharma-db.sh >> /var/log/pharma-backup.log 2>&1") | crontab -
```

9.4 Clean Up Old Images

```
# Remove unused images
sudo docker image prune -a

# Remove stopped containers
sudo docker container prune

# Remove unused volumes
sudo docker volume prune

# Remove unused networks
sudo docker network prune
```

9.5 SSL Certificate Renewal

Certbot auto-renews certificates. To manually renew:

```
# Test renewal
sudo certbot renew --dry-run

# Force renewal
sudo certbot renew --force-renewal

# Check certificate expiry
sudo certbot certificates
```

9.6 Monitor Server Resources

```
# Check disk usage
df -h

# Check memory usage
free -h

# Check CPU usage
top

# Check Docker disk usage
sudo docker system df
```

---

10. Troubleshooting

10.1 Container Won't Start

Problem: Container fails to start or exits immediately

Solution:

```
# Check container logs
sudo docker-compose logs frontend

# Check if port is already in use
sudo netstat -tulpn | grep 5198

# Remove and recreate container
sudo docker-compose down
sudo docker-compose up -d frontend
```

10.2 502 Bad Gateway Error

Problem: Nginx shows "502 Bad Gateway"

Solution:

```
# Check if container is running
sudo docker ps

# Check Nginx configuration
sudo nginx -t

# Check Nginx error logs
sudo tail -f /var/log/nginx/error.log

# Verify proxy_pass port matches container port
sudo grep -r "proxy_pass" /etc/nginx/sites-available/

# Restart Nginx
sudo systemctl restart nginx
```

10.3 Docker Permission Denied

Problem: "Permission denied" when running Docker commands

Solution:

```
# Add user to docker group
sudo usermod -aG docker $USER

# Log out and log back in, or run:
newgrp docker

# Alternative: Use sudo
sudo docker ps
```

10.4 Image Pull Failed

Problem: Cannot pull image from Docker Hub

Solution:

```
# Login to Docker Hub
sudo docker login

# Verify repository exists
# Visit: https://hub.docker.com/r/abdullahdkc/pharma-docker

# Check if image tag is correct
sudo docker-compose config

# Pull manually
sudo docker pull abdullahdkc/pharma-docker:frontend
```

10.5 Backend API Connection Issues

Problem: Backend cannot connect to SQL Server

Solution:

```
# Check backend logs
sudo docker logs backend

# Common error: "Login failed for user"
# Fix: Verify database credentials in .env file
sudo nano /var/www/htdocs/pharma/.env

# Common error: "Cannot connect to server"
# Fix: Check SQL Server is running
systemctl status mssql-server

# Test connection from backend container
sudo docker exec -it backend sh
# Inside container:
apk add --no-cache curl
curl -v telnet://172.17.0.1:1433
exit

# Verify DATABASE_URL format
echo $DATABASE_URL
# Should be: sqlserver://172.17.0.1:1433;database=embellish_prod;user=pharma_user;password=...;encrypt=true;trustServerCertificate=t

# Restart backend after fixing
sudo docker-compose restart backend
```

10.6 Prisma Migration Errors

Problem: Prisma migrations fail during deployment

Solution:

```
# Check Prisma status
cd /var/www/htdocs/pharma/backend
npx prisma migrate status

# Reset migrations (CAUTION: Deletes all data)
npx prisma migrate reset --force

# Apply migrations manually
npx prisma migrate deploy

# If migration is out of sync
npx prisma migrate resolve --applied "<migration_name>"

# Generate Prisma Client
npx prisma generate

# Check database schema
sqlcmd -S localhost -U pharma_user -P "Pharma@SecurePass2026" -d embellish_prod -C
SELECT * FROM _prisma_migrations;
GO
EXIT
```

10.7 Backend Container Logs Show Errors

Problem: Backend container exits or shows errors

Solution:

```
# View full logs
sudo docker logs backend --tail 100

# Common errors and fixes:

# Error: "Cannot find module '@prisma/client'"
sudo docker exec backend npm install @prisma/client
sudo docker-compose restart backend

# Error: "Port 4000 already in use"
sudo netstat -tulpn | grep 4000
sudo kill -9 <PID>
sudo docker-compose restart backend

# Error: "JWT_SECRET is not defined"
# Check .env file has JWT_SECRET
grep JWT_SECRET /var/www/htdocs/pharma/.env

# Run backend in debug mode
sudo docker-compose stop backend
sudo docker run -it --rm -p 4000:4000 --env-file .env abdullahdkc/pharma-docker:backend
```

10.8 GitHub Actions Fails

Problem: CI/CD workflow fails in GitHub Actions

Common Causes & Solutions:

Build Failure:

- Check GitHub Actions logs for TypeScript errors
- Fix errors locally and push again

Docker Push Failure:

- Verify `DOCKER_PASSWORD` secret is correct
- Ensure Docker Hub repository exists and is accessible

SSH Deployment Failure:

- Verify all secrets are configured correctly
- Test SSH key manually:

```
ssh -i github_actions_key deployer@72.61.170.243
```

10.6 SSL Certificate Issues

Problem: SSL certificate errors or expiration

Solution:

```
# Check certificate status
sudo certbot certificates

# Renew certificate
sudo certbot renew --force-renewal

# Restart Nginx
sudo systemctl restart nginx
```

10.7 Application Shows Old Version

Problem: Deployed application shows old code after update

Solution:

```
# Force pull latest image
sudo docker-compose pull frontend --no-cache

# Remove old container and create new
sudo docker-compose down
sudo docker-compose up -d frontend

# Clear browser cache
# In browser: Ctrl+Shift+R (hard refresh)
```

10.8 Container Keeps Restarting

Problem: Container status shows "Restarting"

Solution:

```
# Check logs for errors
sudo docker logs frontend

# Common causes:
# 1. Port conflict - change port in docker-compose.yml
# 2. Configuration error - check environment variables
# 3. Application crash - review application logs

# Temporary: Stop auto-restart to debug
sudo docker update --restart=no frontend
```

Quick Reference Commands

Git Commands

```
git status                      # Check repository status
git add .                       # Stage all changes
git commit -m "message"         # Commit changes
git push origin master          # Push to GitHub
git pull origin master          # Pull latest changes
```

Docker Commands

```
docker ps                       # List running containers
docker images                   # List images
docker logs <container>         # View logs
docker exec -it <container> sh  # Enter container shell
docker-compose up -d            # Start containers
docker-compose down             # Stop containers
docker-compose restart          # Restart containers
```

Nginx Commands

```
sudo nginx -t                   # Test configuration
sudo systemctl status nginx     # Check status
sudo systemctl restart nginx    # Restart Nginx
sudo systemctl reload nginx     # Reload configuration
```

Server Commands

```
df -h                          # Disk usage
free -h                        # Memory usage
top                            # CPU usage
sudo systemctl status docker   # Docker service status
```

SQL Server Commands

```
# Check SQL Server status
systemctl status mssql-server

# Restart SQL Server
sudo systemctl restart mssql-server

# Connect to SQL Server
sqlcmd -S localhost -U pharma_user -P "password" -d embellish_prod -C

# List databases
SELECT name FROM sys.databases;
GO

# List tables
USE embellish_prod;
GO
SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE='BASE TABLE';
GO
```

Prisma Commands

```
# Generate Prisma Client
npx prisma generate

# Run migrations
npx prisma migrate deploy

# Check migration status
npx prisma migrate status

# Open Prisma Studio (database GUI)
npx prisma studio

# Reset database (development only)
npx prisma migrate reset
```

Contact Information

Technical Support:

- Repository: https://github.com/wizone-dev/embellish
- Docker Hub: https://hub.docker.com/r/abdullahdkc/pharma-docker
- Production URL: https://pharma.wizoneit.com

Server Details:

- VPS IP: 72.61.170.243
- SSH User: deployer
- Domain: pharma.wizoneit.com

Document History

Version	Date	Changes	Author
1.0	Jan 6, 2026	Initial frontend-only deployment documentation	GitHub Copilot
2.0	Jan 6, 2026	Added complete backend deployment with Node.js Express, MS SQL Server, Prisma ORM	GitHub Copilot

---

Appendices

A. Dockerfile - Frontend

```
# Build stage
FROM node:20-alpine as build

WORKDIR /app

COPY package*.json ./
RUN npm ci

COPY . .
RUN npm run build

# Production stage
FROM nginx:alpine

COPY --from=build /app/dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

B. Dockerfile - Backend

```
FROM node:20-alpine

WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

COPY . .
RUN npm run build

EXPOSE 4000

CMD ["node", "dist/index.js"]
```

C. Complete docker-compose.yml (Full Stack with MS SQL)

```
version: "3.8"

services:
  backend:
    image: abdullahdkc/pharma-docker:backend
    container_name: backend
    restart: always
    ports:
      - "4000:4000"
    environment:
      - NODE_ENV=${NODE_ENV}
      - PORT=${PORT}
      - DATABASE_URL=${DATABASE_URL}
      - JWT_SECRET=${JWT_SECRET}
      - JWT_EXPIRES_IN=${JWT_EXPIRES_IN}
      - CORS_ORIGIN=${CORS_ORIGIN}
    extra_hosts:
      - "host.docker.internal:172.17.0.1"
    networks:
      - pharma-network

  frontend:
    image: abdullahdkc/pharma-docker:frontend
    container_name: frontend
    restart: always
    ports:
      - "5198:80"
    depends_on:
      - backend
    environment:
      - VITE_API_URL=${VITE_API_URL}
    networks:
      - pharma-network

networks:
  pharma-network:
    driver: bridge
```

End of Document